Sequential circuits

مقدمه :-

الدوائر الالكترونية منها نوعين هم (sequential and combinational)

الدوائر ال combinational دي زي (multiplexer ,adder ,not ,or ,and,....) وكلها دوائر لا يعتمد فيه الlutput علي ال Onput , عشان كده مش محتاجة ميموري علشان تخزن اوتبوت سابق

اما الدوائر ال sequential دي زي ال register ودي دواير يعتمد فيها الانبوت علي الاوتبوت علشان كده بيكون فيها ميموري لتخزين اخر اوتبوت لاستخدامه ك input تاني

so we use both sequential and combinational circuits in computer making

Synchronous - In this mode, all circuits in the system change their state at some precisely defined instants

لما بيحصل تغيير في الدايره بتلاقي كل العناصر متزامنه في التأثر بالتغيير ده

Asynchronous - The time at which a change occurs in one circuit has no relation to the time a change occurs in another circuit (not synchronized)

(لو حصل تغير في جزء من الدايره فده مش هيكون متزامن مع حدوث تغير في جزء اخر من الدايره)

Digital circuits can operate in asynchronous or synchronous mode

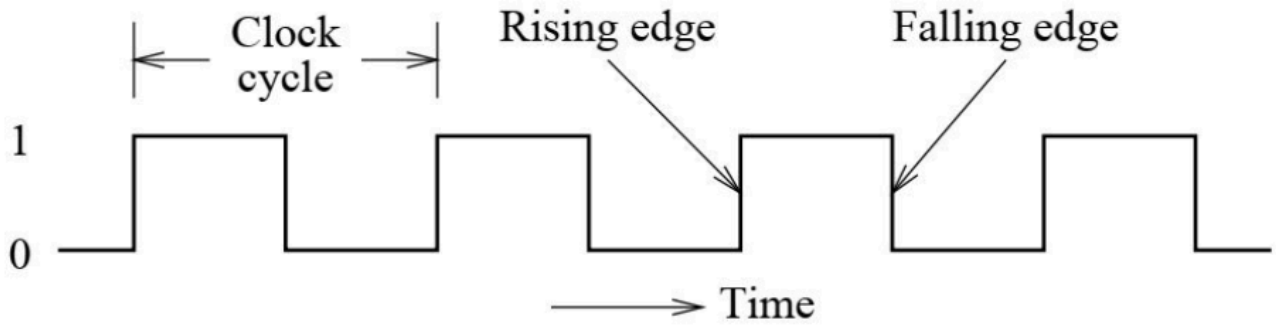
> **"Note"**
> **علشان اعمل الدايرة synchronous**
>
> **لازم يبقي عندي clock ولازم الclock دي تكون متوصلة بجميع اجزاء الدايره سواء بشكل مباشر او غير مباشر**

• The clock signal in a digital system provides such a global definition of time instants at which changes can take place.

• The clock signal also specifies the speed at which a circuit can operate.

يتم تمثيل الclock بقمم (وحايد on) وقيعان (اصفار off) وبيكون ليها زمن دوري

(clock cycle or time period or frequency) وتقاس بوحده (pulse /second) او الهيرتز Hz



***rising edge-positive edge-(from 0 to 1)**

***falling edge-negative edge-(from 1 to 0)**

• A clock cycle/period "T" is defined as the time between two successive rising (or falling) edges

> "clock existence in digital circuits provides :
> — The global synchronization signal for the entire system
>
> — Timing information in the form of a clock period"

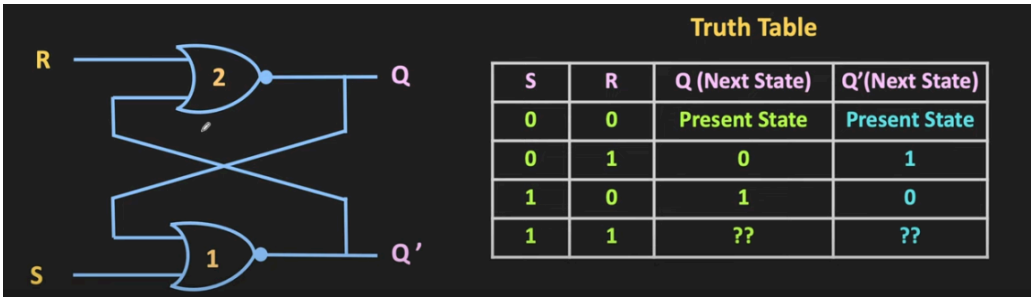
Sequential elements :-(latch & flip flop)

^latch and flip flop are the basic elements of the sequential circuits

^they represent 1 bit memory

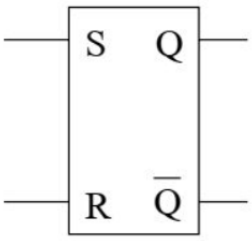
> "Focus"
> ***latch is level- sensitive device which responds to high or low signal**
>
> ***flip flop is edge-triggered (output changes only at rising or falling edge**

~S-R latch(set and reset using nor gates):



Truth Table

S	R	Q (Next State)	Q'(Next State)
0	0	Present State	Present State
0	1	0	1
1	0	1	0
1	1	??	??



نظرا لان الانبوت هو (الs و ال r و اخر اوتبوت) اول حالة في الجدول لا يمكن تحديد الناتج لان منعرفش اخر اوتبوت كان كام والحالة دي اسمها NO CHANGE

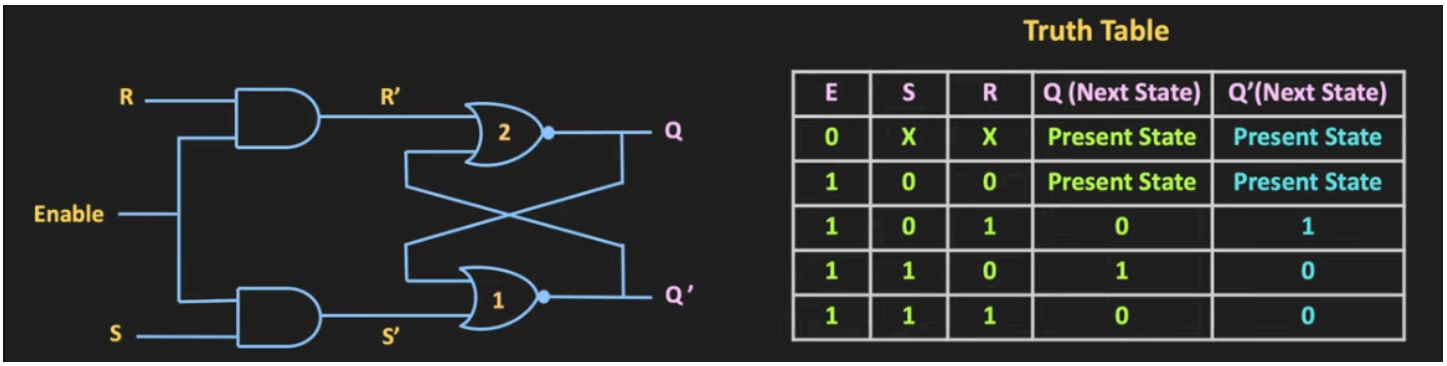
تاني حالة في الجدول اسمها reset نظرا لان ال r بواحد وتالت حالة اسمها set والرابعة hazard

the fourth state(1,1) is avoidable

> **notes ":**
> **Latch has the capability to store a bit**
>
> **• To write 1 into this latch, set S as 1 and R as 0(SR =10)**
>
> **• To write 0, use S=0 and R = 1(SR=01)**
>
> **• To retain a stored bit, keep both S and R inputs at 0(no change)**
>
> **• Avoid setting SR to 11 (hazard)**
>
> **• Once we have the design to store a single bit, we can**
>
> **replicate this circuit to store more bits or bytes.**

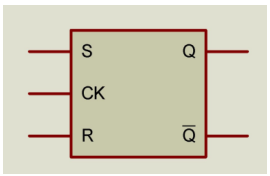
to avoid Asynchronous in S-R latch , we add clock to her and it becomes clocked S-R latch

~ clocked S-R latch



Truth Table

E	S	R	Q (Next State)	Q'(Next State)
0	X	X	Present State	Present State
1	0	0	Present State	Present State
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

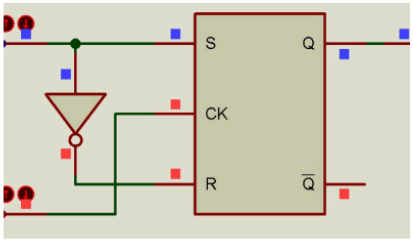


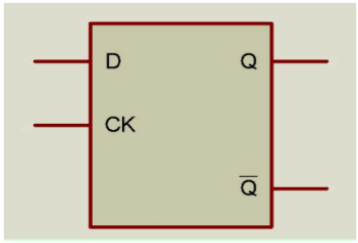
A problem with SR latches is that we have to avoid the SR = 11 input combination

• This problem is solved by the D latch

~D-latch

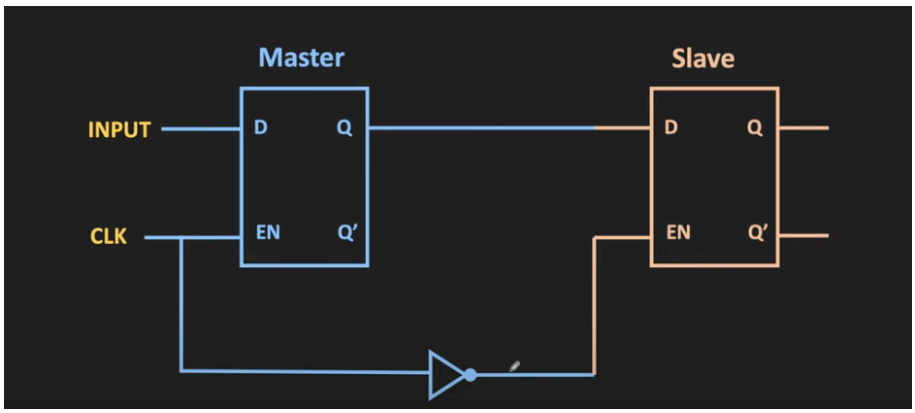
هخليهم جايين من r input و s input وعندي (2signals) فكرتها ان بدل ما يبقي (1signal)



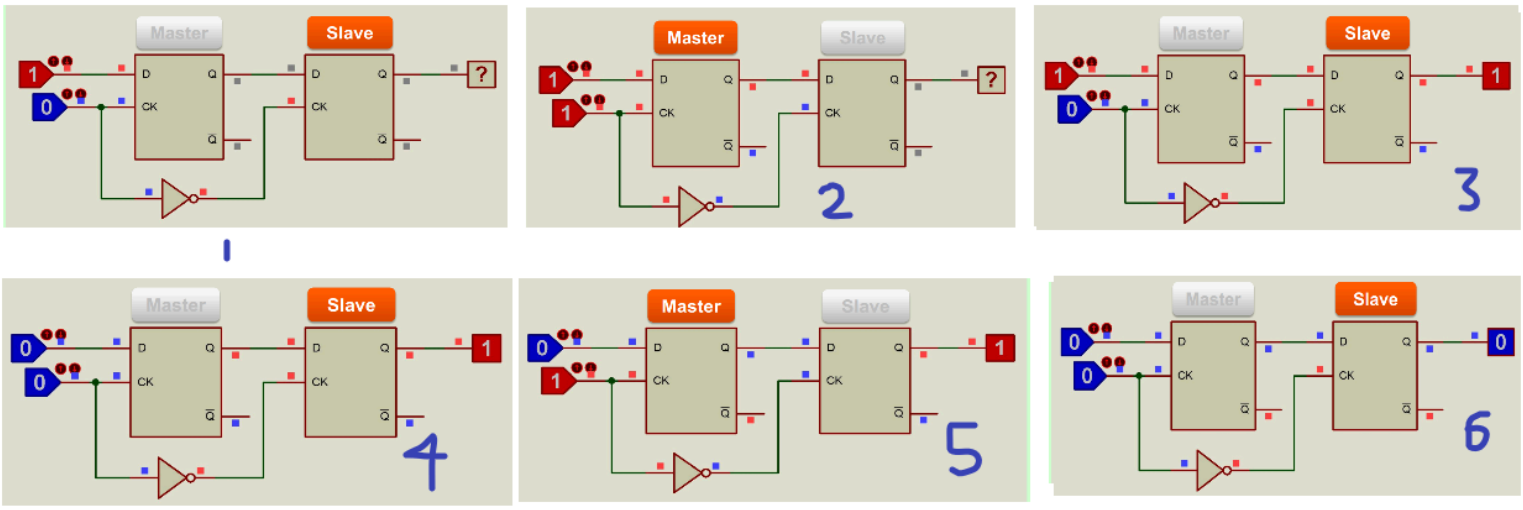


D	Q_{n+1}
0	0
1	1

~Master-Slave D flip flop

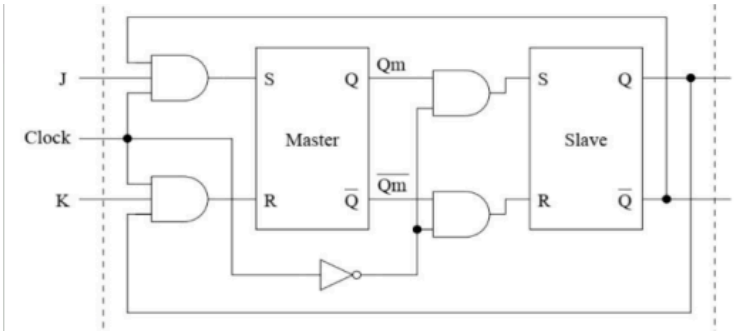


لو ال clk بصفر هيكون ال master داخل لها صفر و ال slave يدخلها واحد فتشتغل ال slave وتقف ال master (والعكس صحيح)

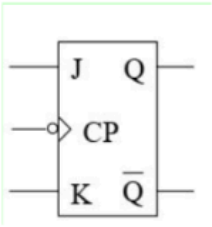


~ J-K flip flop

- **It is another type of Flip-Flops that makes the avoided input combination “SR”=11 useful**
- **J-K Flip-Flop uses a 2nd feedback**
- **هي عباره عن negative edge triggered SR flip flop متوصل معاها الK و j عن طريق and gates**

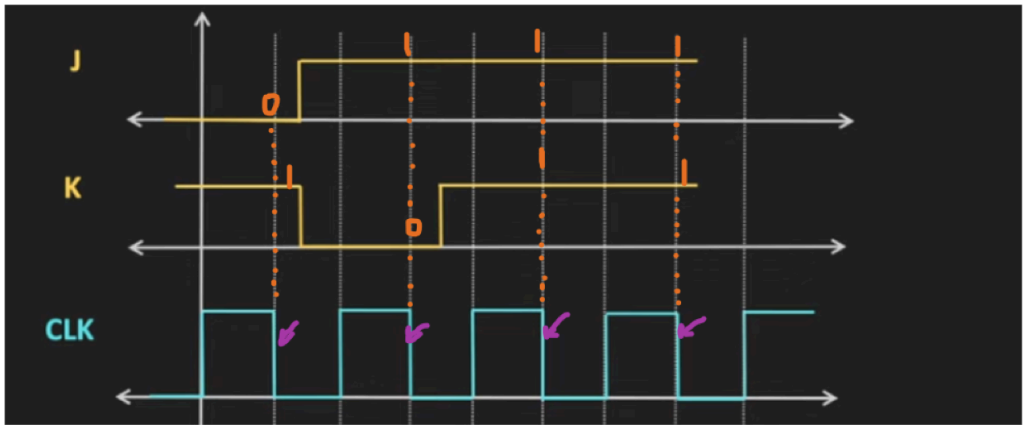


J	K	Q_{n+1}	
0	0	Q_n	no change
0	1	0	reset
1	0	1	set
1	1	$\overline{Q}_n$	toggle



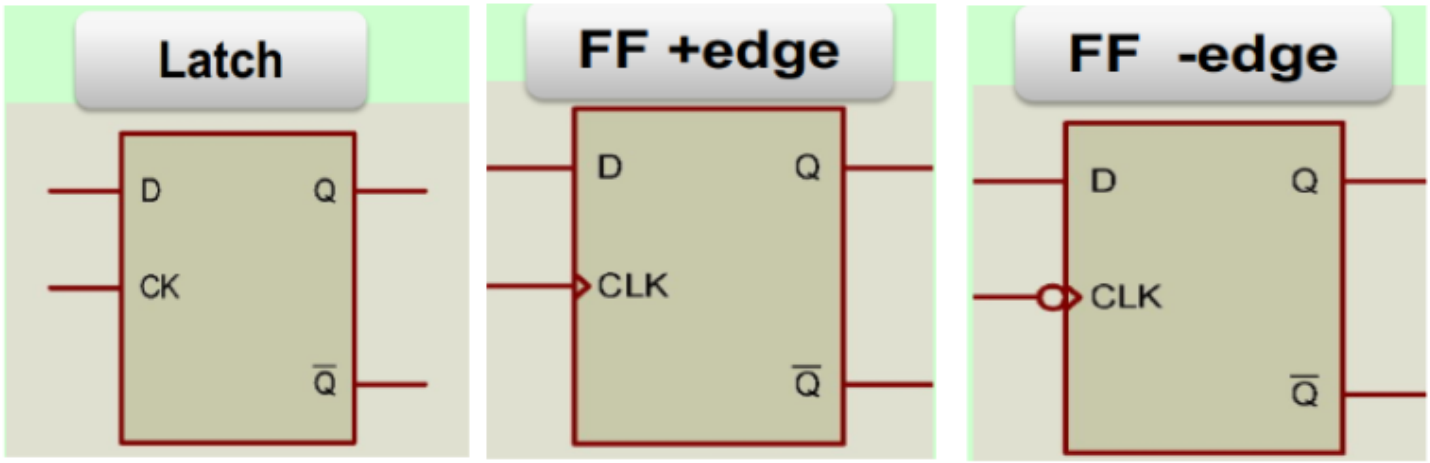
> **“notice”**
> **— J-K works as SR flip flop in the the first three states(00,01,10,)**
>
> **but the last state(11) is a toggle as it makes the next state =the reverse of current state(Qn`=Q(n+1))**
>
> **_ the state (11)became determined here**

As it is a negative edge triggered , we determine the output in the falling edge of the clock(عليه سهم لونه موف)



note
D latch avoids state(11) but JK flip flop makes this state useful

notice the difference between them ⬇:



both latch and flip flop can store 1 bit so we need many of them to store data so we use registers

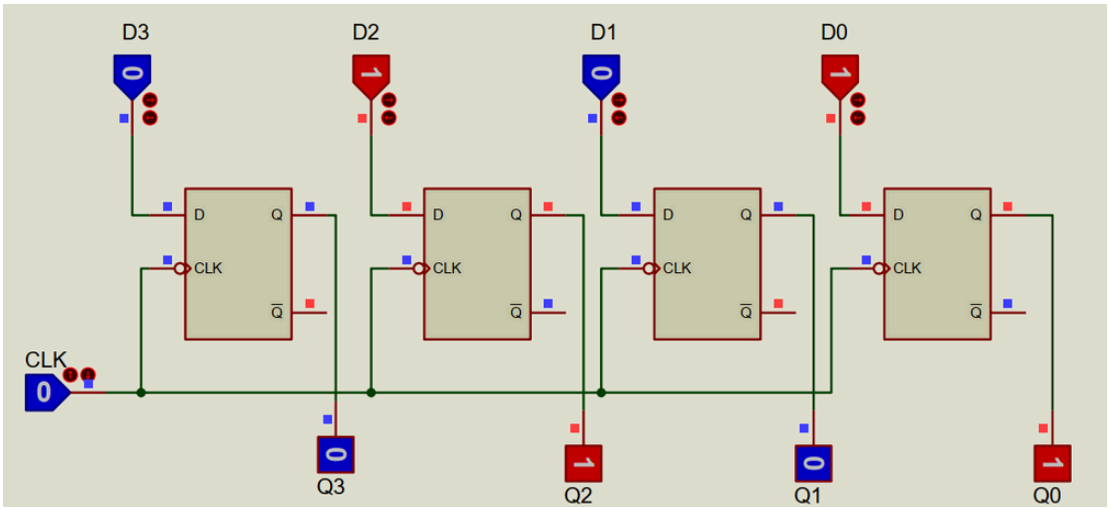
- **Register is used for temporary data storage in all CPUs**
- **They are used for data storage based on Flip Flop so it consists of one or more FF**
- **Mainly there are two types of Registers : Parallel Registers —Serial Registers**

~parallel register ⬇

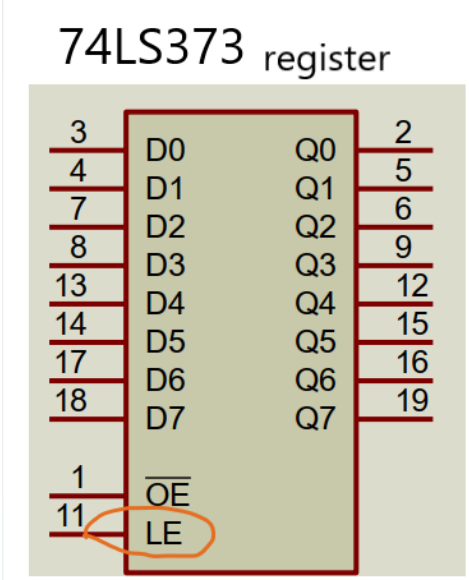
it stores all data bits at the same time and needs one clock

(4-bit parallel register is a form of parallel registers which stores 4 bits and consists of 4 D flip flop)

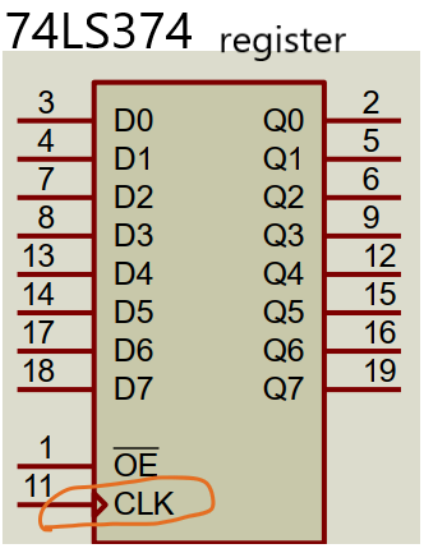
وبفرض ان الاربعة FF دول كانوا (negative edge triggered) فهيشتغلوا وقت تغيير ال clk من 1 الي صفر



practical examples for parallel registers ⬇



Octal transparent latch with 3-state output



Octal D-type FF with tri-state output

note : LE (latch enable) usually said with latches but CLK usually said with flip flop

OE means output enable.

Sara Ibrahim

Note (:

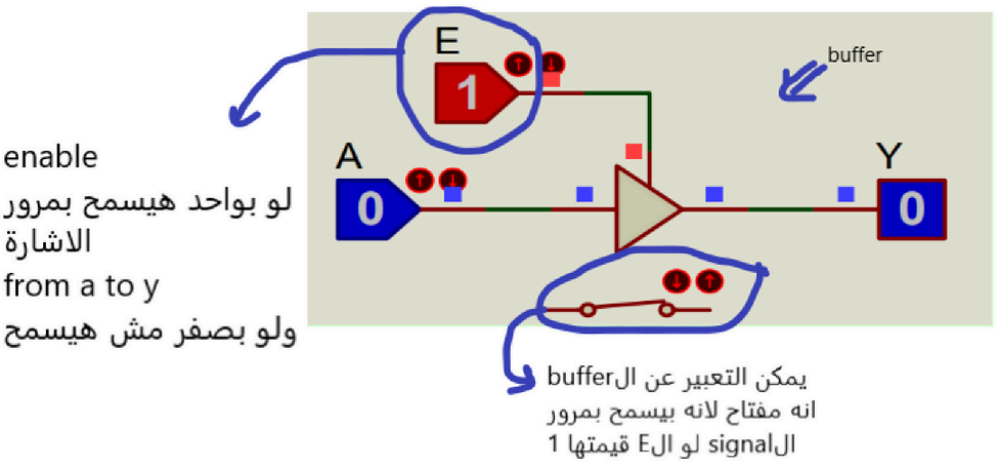
The digital buffer is the logic gate opposite of an inverter (Not Gate) we look at in the previous tutorial where we saw that the NOT gates output state is the complement, opposite or inverse of its input signal.

sometimes in digital electronic circuits we need to isolate logic gates from each other or have them drive or switch higher than normal loads without the need for inversion. One type of single input logic gate that allows us to do just that is called the **Digital Buffer**.

شكل ال buffer بيبقي شبه ال not gate من غير دايره صغيره

the signal which comes from output is weak so we use buffer to generate stronger signal(يعني بقدرة اعلي) from the original signal

Tri - state buffer (:



Logic circuits are two state system 0/1 (High-Low)

• 3-state means the system will have additional state to

the two state system (High-Low – High impedance) ⬇

high : if E =1 & A=1 ,, low : if E=1 & A =0 ,,

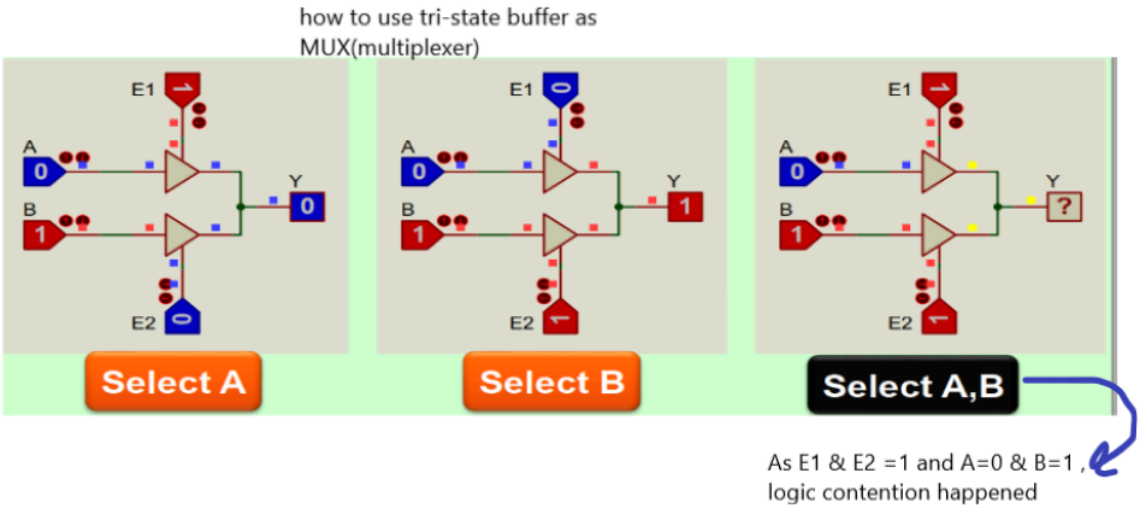
high impedance : if E = zero (الناتج مقاومة عالية لانها تعتبر دايره مفتوحه)

• A tri-state buffer replicates the input if E = 1

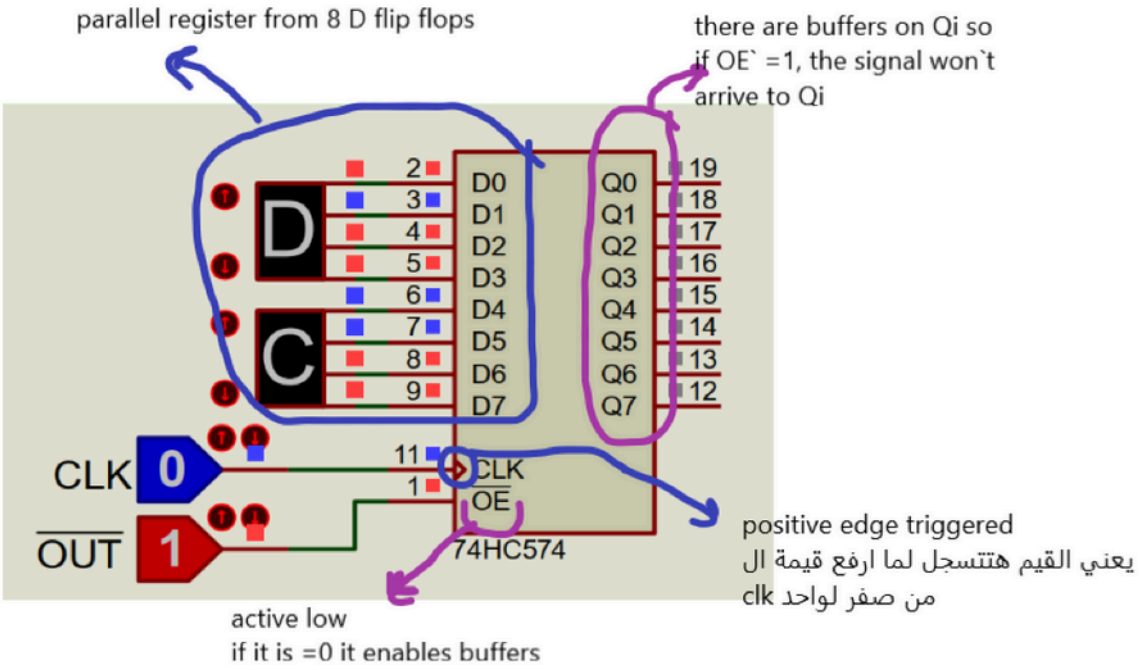
If E = 0 the output of the buffer is disconnected or open circuited

Tri-state buffer can be used to implement data multiplexing (Multiplexer)

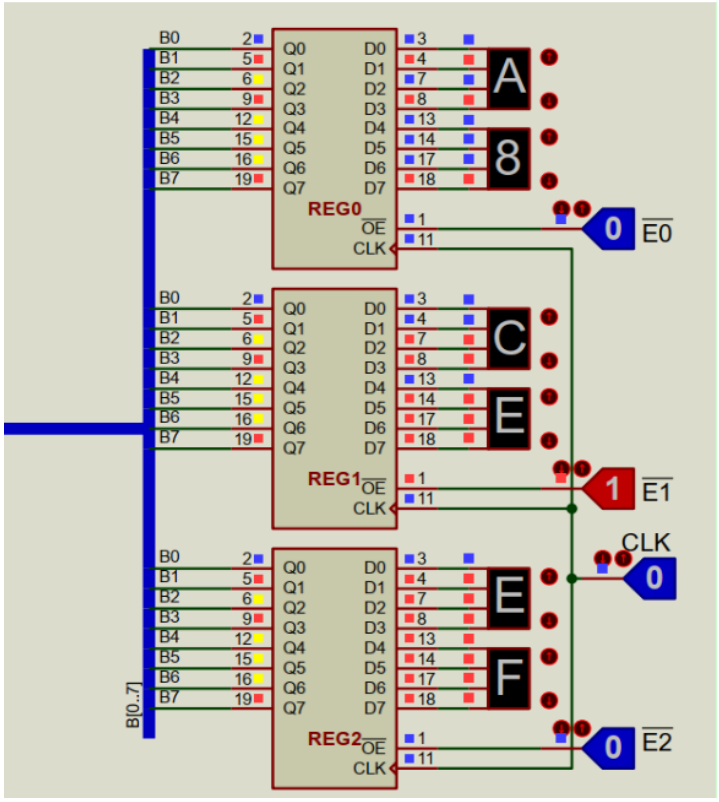
• Multiplexer is an essential part in the design of CPUs



74LS374 register ⬇



Bus MUX using tri-state buffers



1-three registers
2-each register has its own enable (E`-active low-)
3- the three has common clock
4- all these registers are configured from 8 D flip flops
5- all these D flip flops are positive edge triggered
يعني هيسجل البت لما نرفع الكلوكمن صفر لواحد
6-each register has its own buffers so it works when its E` =0
في الرسمه دي تم رفع الكلوك من صفر لواحد علشان كده القيم اتسجلت بعدها تم تغيير الكلوك من واحد لصفر فحصل وضع ال no change
7- no output as all E0` &E1` & E2` equal zero
يعني كلهم شغالين فالاوتبوت مش عارف ياخد قيمه من انهي واحد فيهم

الرسمه دي 3 ريجسترز بيعملوا out بس يعني للتوضيح بس (لكن الصح ان كل ريجستر فيهم يعمل in و out)

how the device above works as MUX ⬇

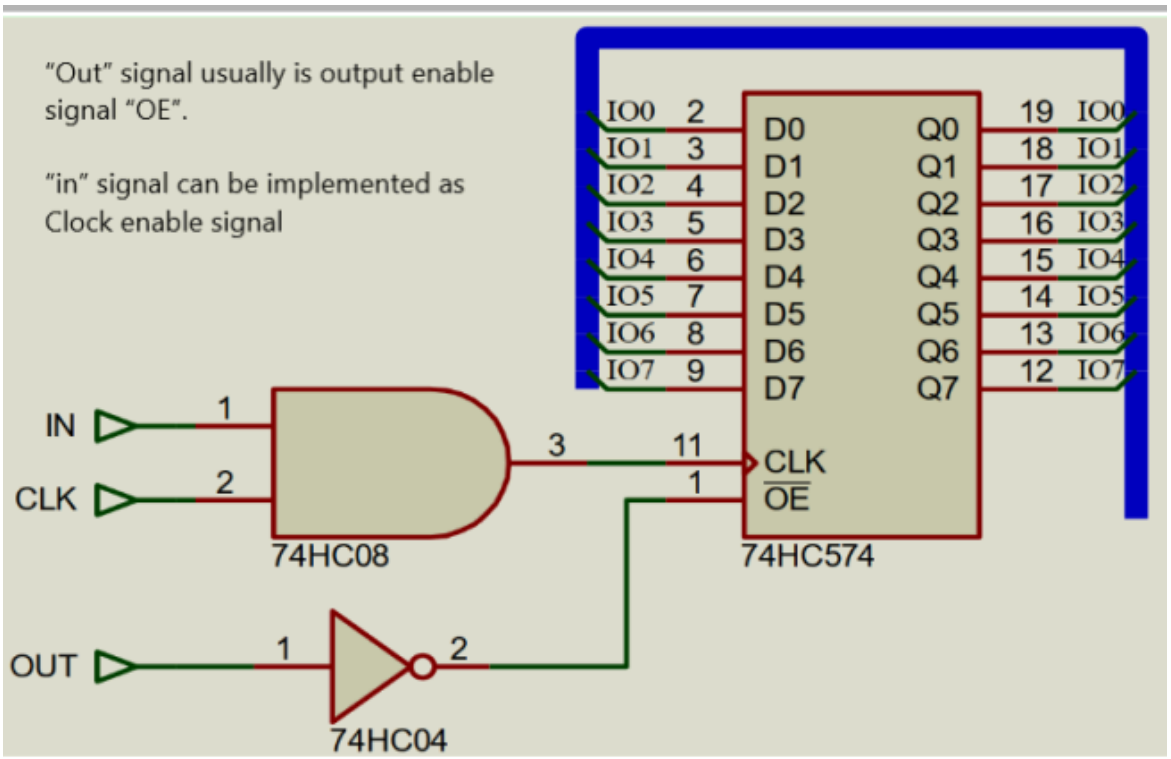
make E` of only one of them to be =0 so the output will takes the value from the register with E`=0

Registers Used in CPU –IN-OUT

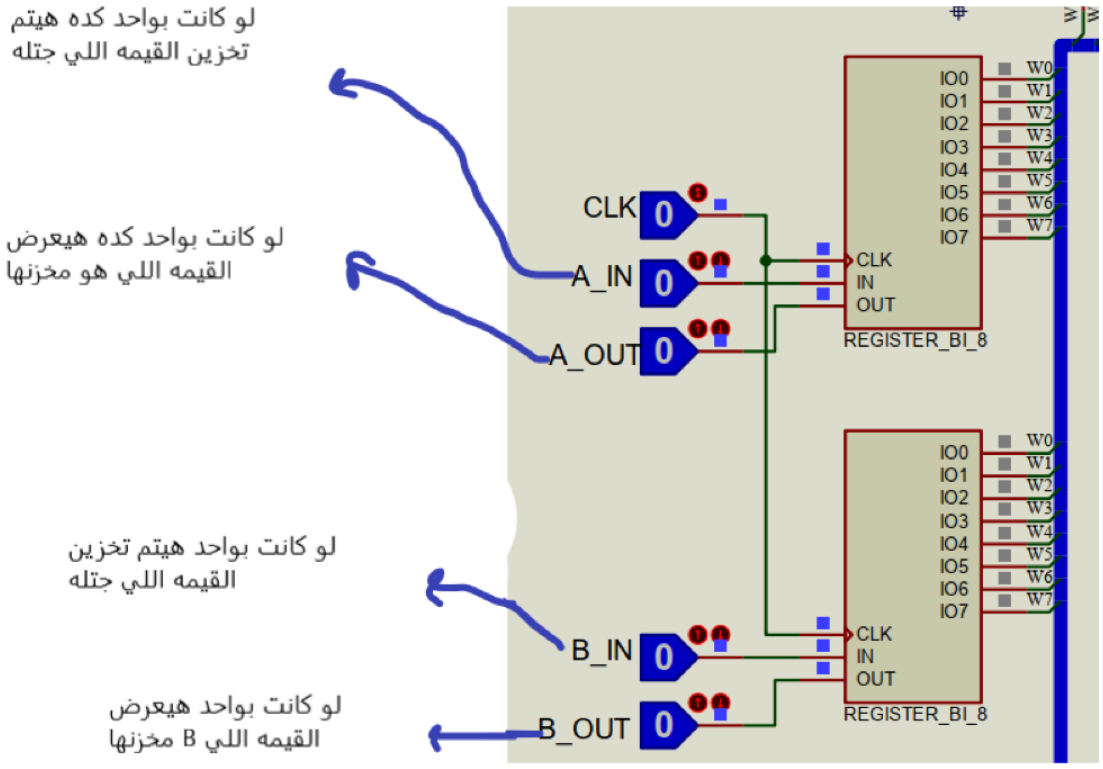
• Registers used in CPU design must have the ability to transfer data to/from other units and this feature is called Register Transfer

• Each Register can do either OUT operation (Read Data) or IN operation (Write Data) so We need a bidirectional (ثنائي الاتجاه) register that can be controlled to in or out data

• A typical implementation of this register consists of D-type FF and Tri-state buffer



ال out في الرسمه هو ال E او OE يعني مفيش داش يعني مش active low

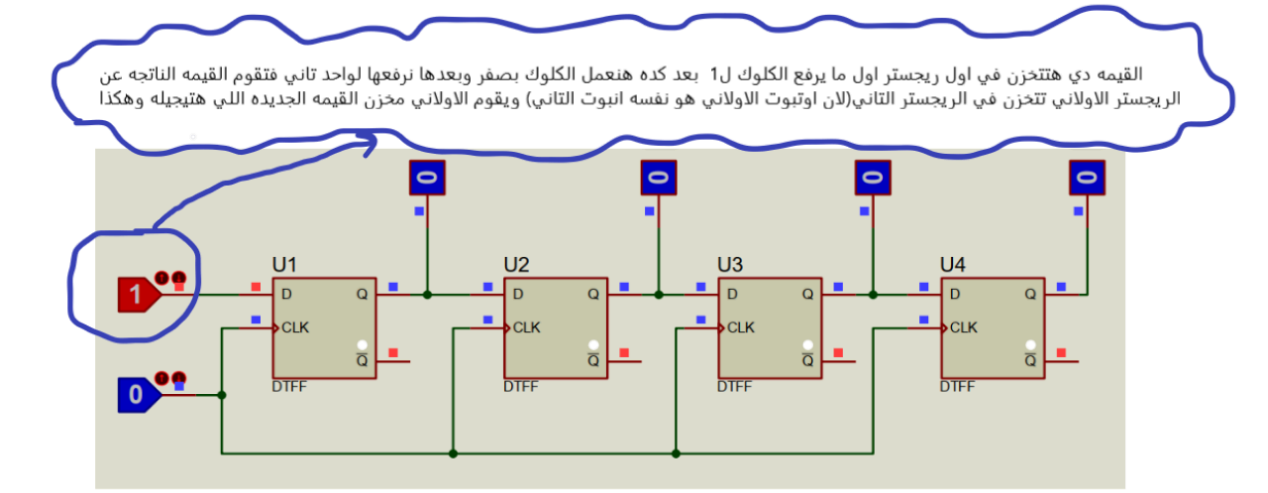


~Serial shift registers ⬇

Serial Registers store data bits one at a time (1-bit per clock cycle)

• (4-bit serial register is a form of serial registers which stores 4 bits and consists of 4 D flip flop connected in cascade)

وبفرض ان الاربعة FF دول كانوا (positive edge triggered) فهيشتغلوا وقت تغيير ال clk من صفر الي 1



وبالشكل ده يكون الريجستر ده شغال serial in - parallel out لان السيجنالز بتدخل ورا بعض وبتخرج مع بعض , وده shift right لان القيم بتروح لليمين ويعتبر ⬇

Serial to Parallel data converter(converts serial data bits to parallel data)

so the shift register can be used as serial in - parallel out

practical examples for serial shift registers ⬇

— Serial to Parallel data converter (اتقال فوق)

— Parallel to Serial data converter

— Counters (especially “step counters”)

---

parallel to serial data converter ⬇

To convert data from parallel to serial, the shift register must be loaded with parallel data, then shifting these data out

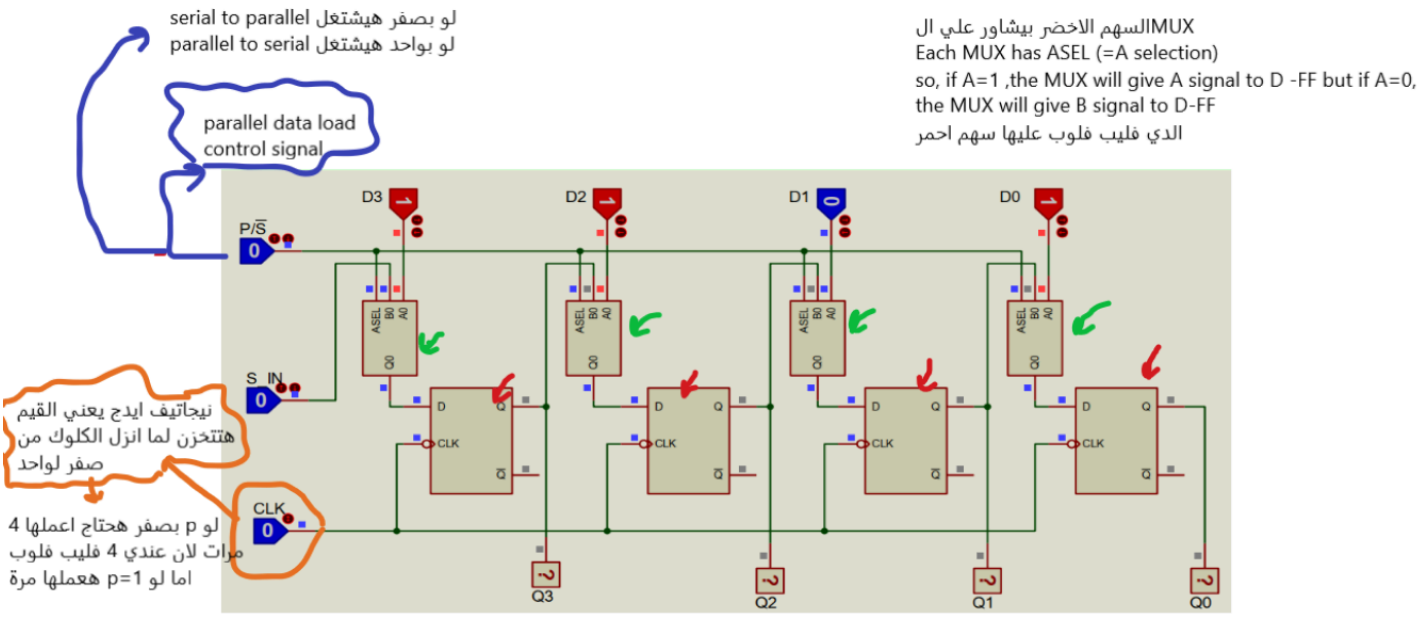
• So this register has 👉 parallel data load control signal

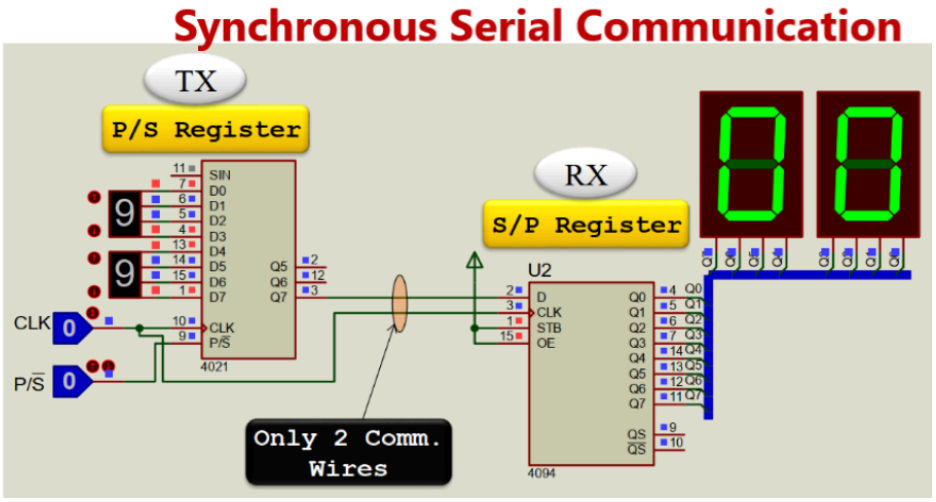
• It can be constructed from D-FFs and MUX

• The D- input for each FF is either connected to

— Data line from parallel input or — Q output from previous FF

• MUX can do this selection





Counters ⬇

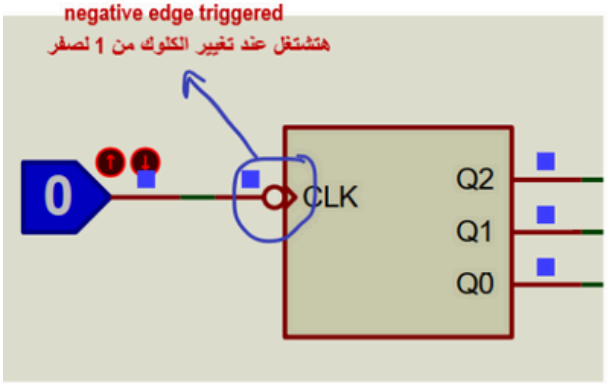
Counter is a sequential logic circuit that has only one input (Clock) and several outputs $Q0$, $Q1$, $Q2$, … .

• With each clock pulse the counter changes its output through different states

• It is called divide by N counter if the sequence of output repeats after N states

• Binary counter produces output states in an incremented or decremented sequence (0, 1 , 2 , 3, 0, 1, 2, 3, …)

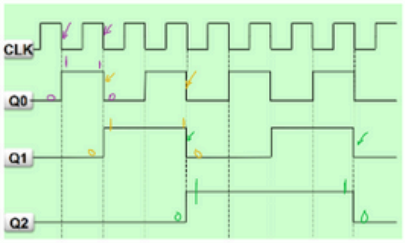
Binary counter (divide by 8)

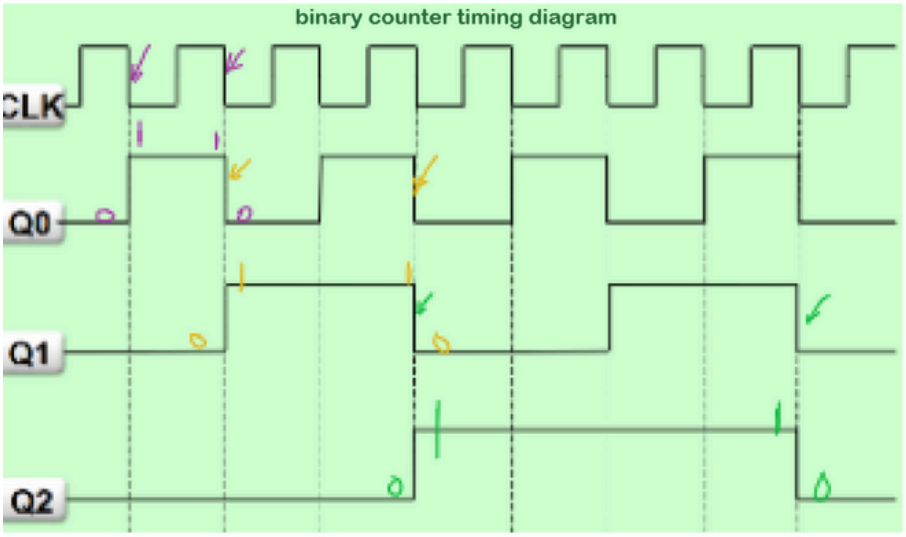


الاحتمالات اللي ممكن تنتج ومتمثلة بيانيا في الرسمه اللي علي اليمين
وهنا N هتساوي 8 (8 احتمالات)والنوتج دي تفضل تتكرر

Q2	Q1	Q0	
0	0	0	= 0
0	0	1	= 1
0	1	0	= 2
0	1	1	= 3
1	0	0	= 4
1	0	1	= 5
1	1	0	= 6
1	1	1	= 7

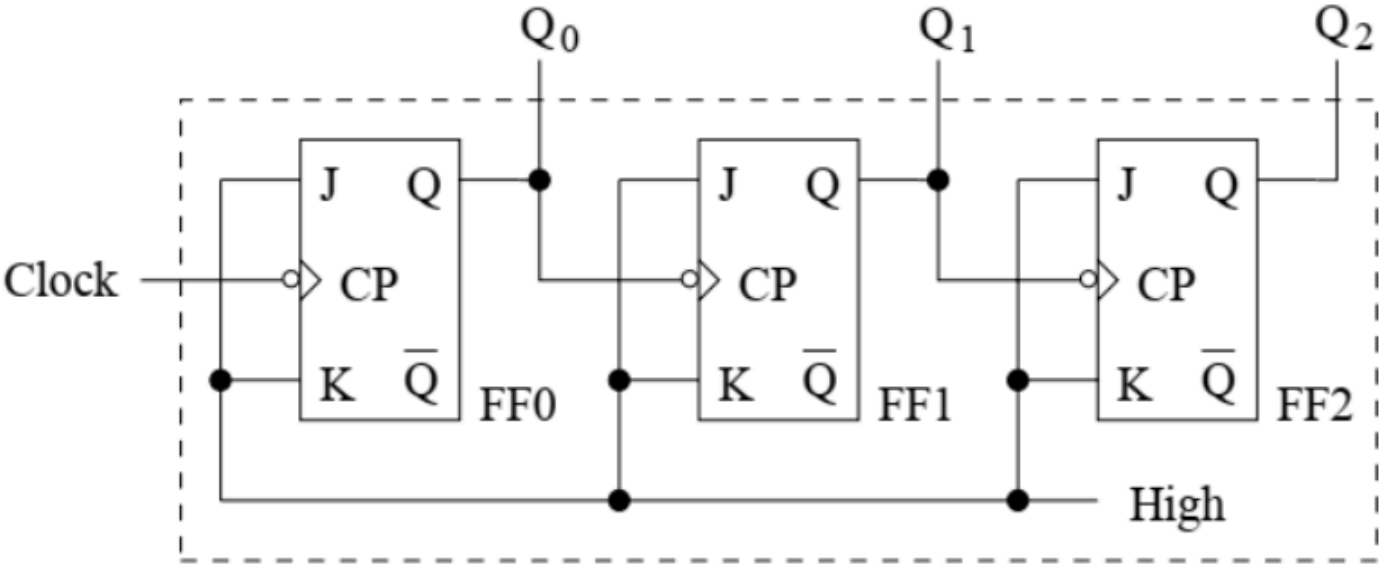
Binary Counter Timing Diagram



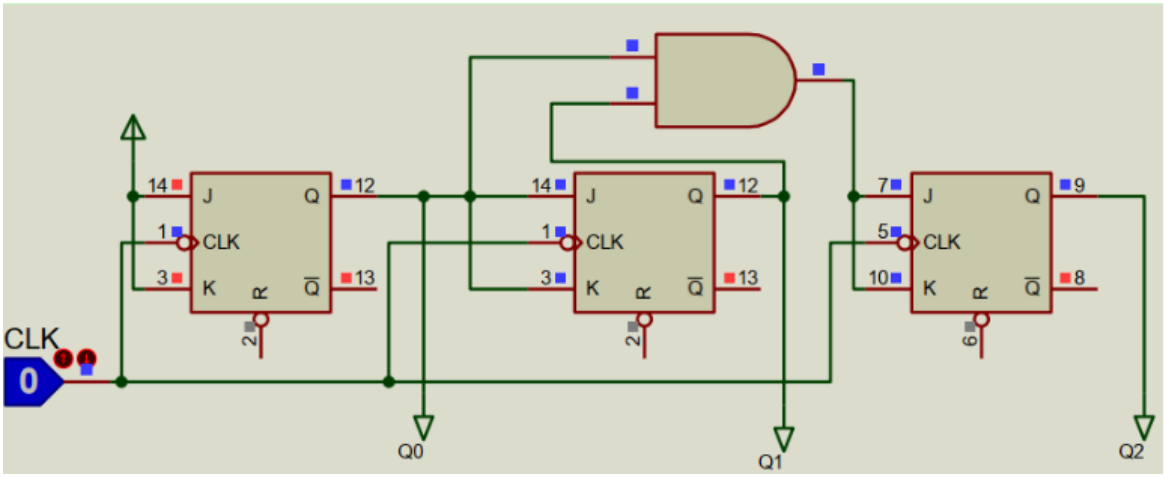


هتلاحظ من الرسم ان :
*كل ما الكلوك تتحول من 1 لصفر بيحصل عكس لقيم Q0 متوضح بالخط الموف
*كل ما Q0 تتحول من 1 بصفر بيحصل عكس لقيم Q1 متوضح بالخط الاصفر
*كل ما Q1 تتحول من 1 لصفر بيحصل عكس لقيم 2Q متوضح بالخط الاخضر
وده معناه ان Q2 معتمده علي Q1 و Q1 معتمده علي Q0 و Q0 معتمده علي الكلوك يعني مش كلهم معتمدين علي الكلوك يعني Asynchronous
علشان كده نقدر نعبر عن ال counter باستخدام J-K ff

- Q0 toggle with each –ve edge of clock
- Q1 toggles with each –ve edge of Q0
- Q2 toggles with each –ve edge of Q1



we need to convert that asynchronous device to synchronous (the clock must be connected to all CLK inputs of the FFs)so this will be like this ⬇

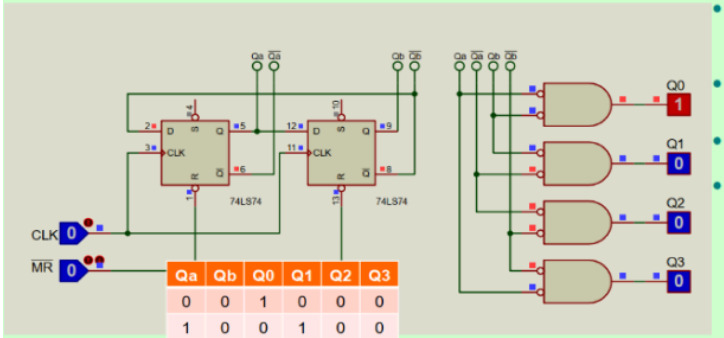


—Q1 toggle with each –ve edge of clock when Q0=1

—Q2 toggles with each –ve edge of clock when both Q0 & Q1=1

Counters using Shift Registers

Step Counter using Shift Register

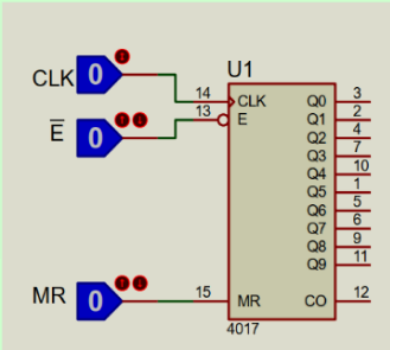


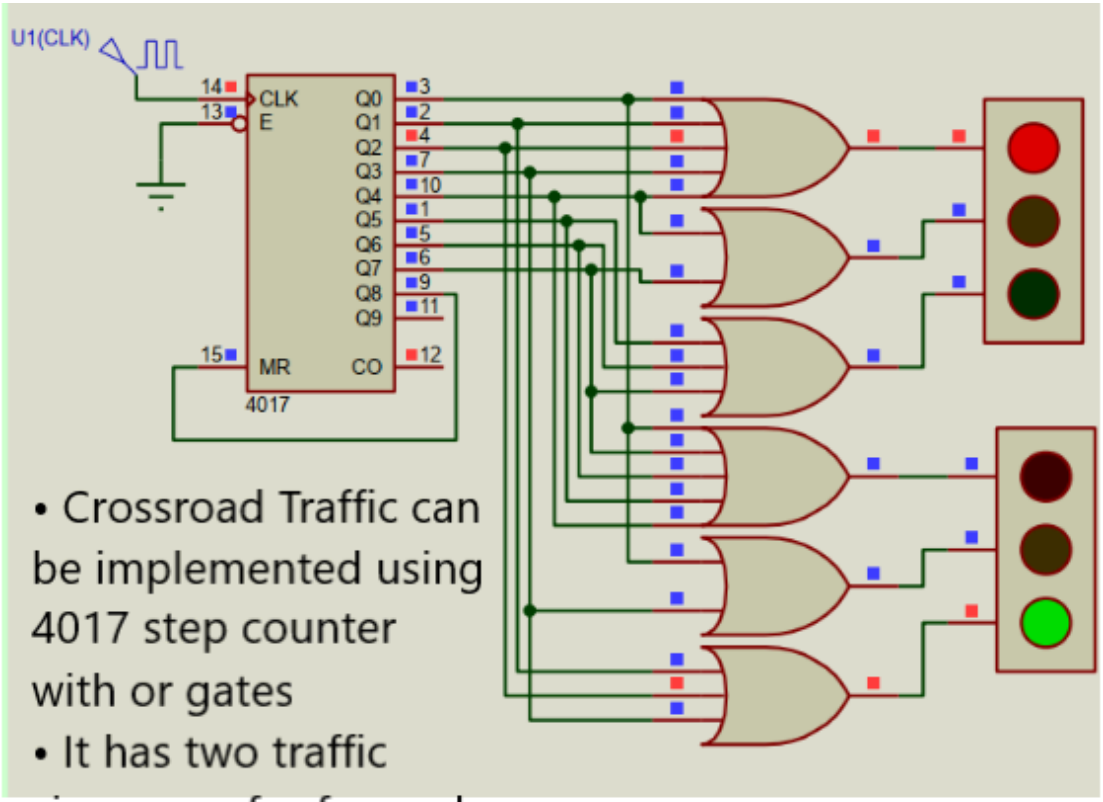
Qa	Qb	Q0	Q1	Q2	Q3
0	0	1	0	0	0
1	0	0	1	0	0
1	1	0	0	1	0
0	1	0	0	0	1

Counters using Shift Registers

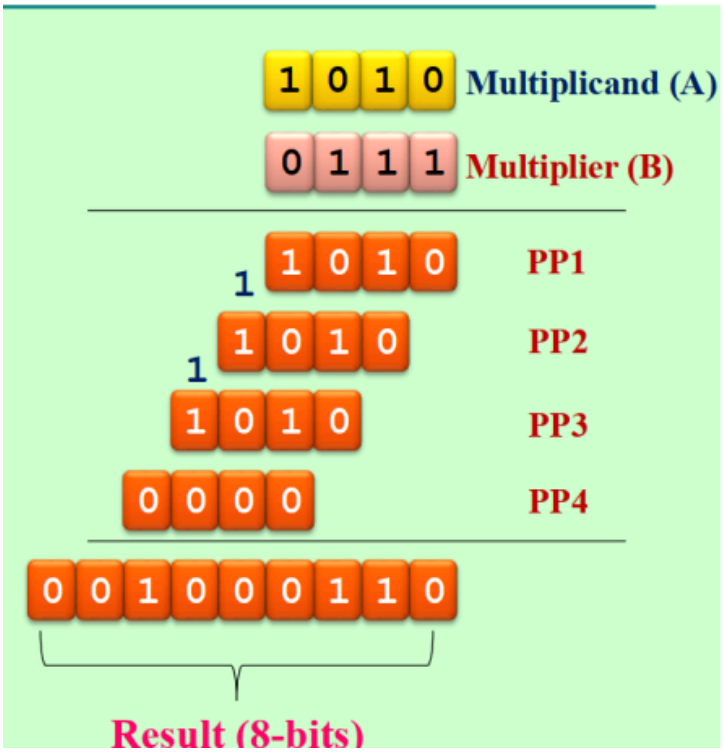
Practical Step Counter "4017"

- It is a decade (÷10) step counter with Reset & Clock Enable
- It can be used as
- Divide by N ≤ 10 counter
- Count to N and stop





- Crossroad Traffic can be implemented using 4017 step counter with or gates
- It has two traffic signs, one for for each direction



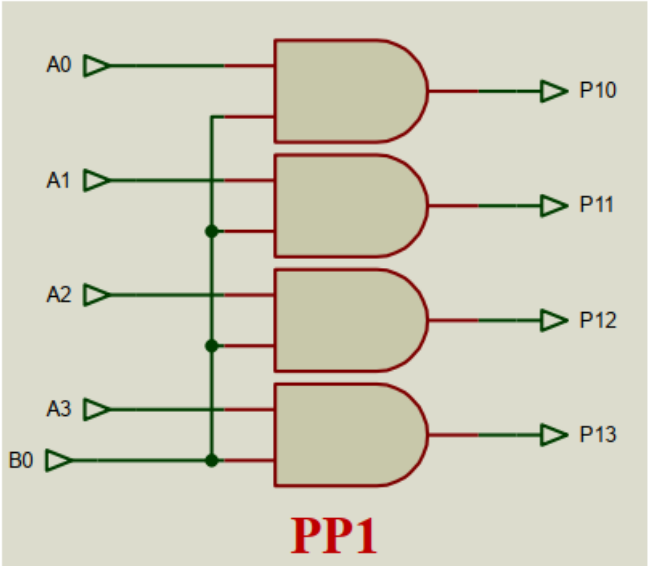
ضرب bit4 في bit4 يبقي الناتج bit 8
بناخد اول بت من b ونضربها في a كلها
في الخطوة اللي بعدها نعمل شيفت مرةونضرب البت التانيه منb في a كلها
بعدها نعمل شيفت مرتين وهكذا

We first compute partial products "*PP(i)*

- if bit(i) in multiplier =1, then *PP(i)* =Multiplicand, else *PP(i) = 0*
- Each *PP(i)* is shifted according to *i*
- Add all *PP(i)* to get the result

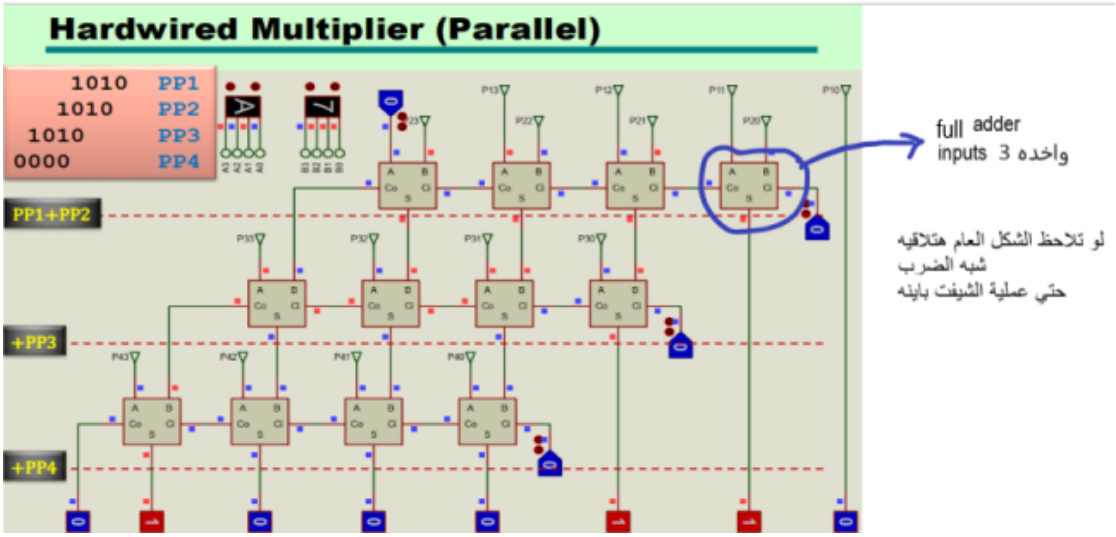
A0 يعني اول بت في A

B0 يعني اول بت في B

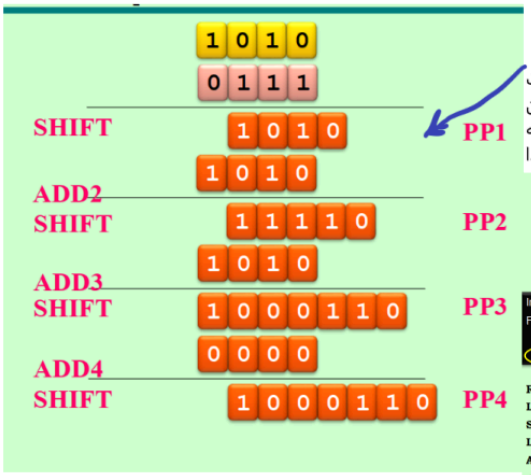


p10 يعني اول بت في p1

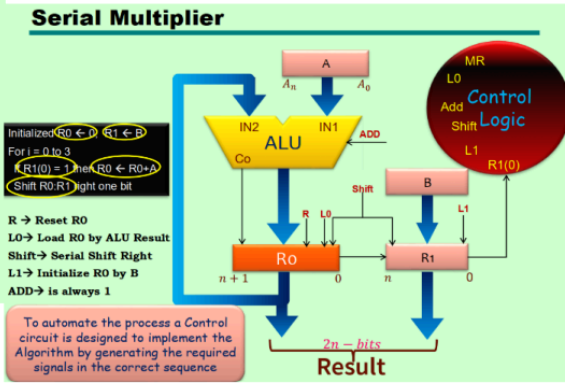
الرسمه دي بتوضح ازاي نفذنا اول خطوه في الضرب باستخدام الجيتس

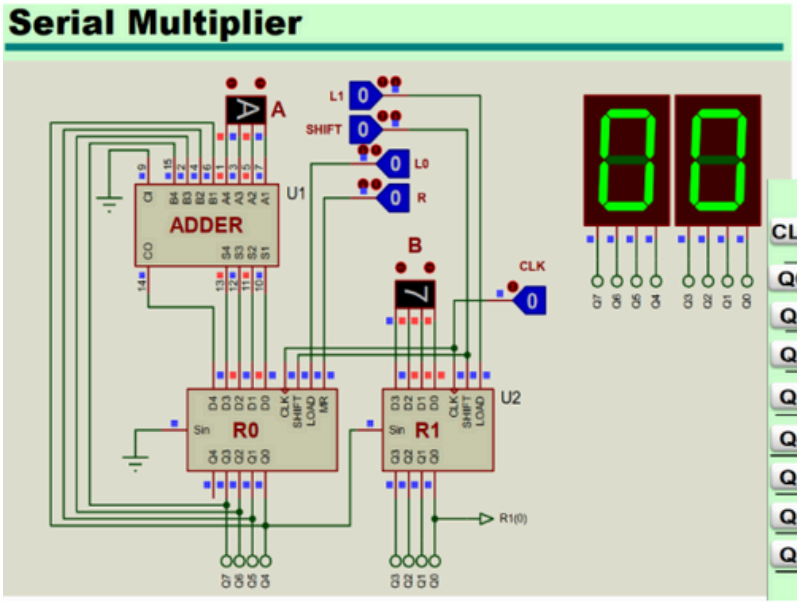


لو تلاحظ الشكل العام هتلاقيه شبه الضرب حتي عملية الشيفت باينه

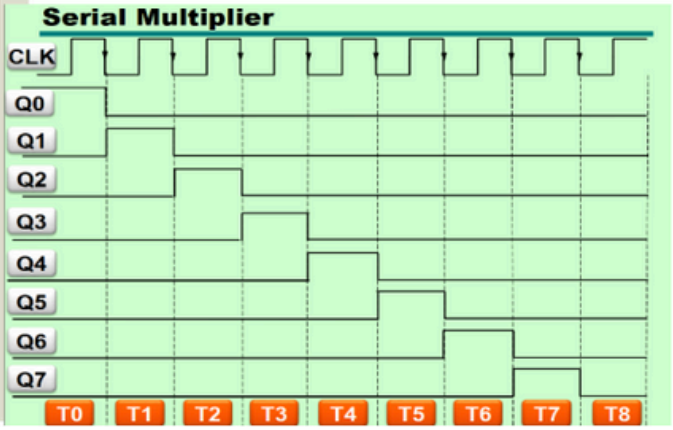


اضرب اول بت من b في a كله بعدها اعمل شيفت للناتج ناحيه اليمين
اضرب تاني بت من b في a كله واجع علي اللي قبله وهكذا


Serial Multiplier
R → Reset R0
L0→ Load R0 by ALU Result
Shift→ Serial Shift Right
L1→ Initialize R0 by B
ADD→ is always 1
To automate the process a Control circuit is designed to implement the Algorithm by generating the required signals in the correct sequence
ALU
Control Logic
Result



How to automate the process?
One solution is to use step counter to produce output signals that represent the time step
Building a combinational logic circuit to produce control signals



++

Chapter 3 👀

before programming , all arithmetic (+,-,*,/) and logical (and , or, not,...) operations on data depended on hardware so the systems were hardwired

ملحوظه : القسمه علي 2و4و8و16و..... تنفع عملية حسابيه او منطقية (both logical and arithmetic)

hardwired systems It is a customized hardware that accepts data and produces results and it is a pure hardware system

• If there is a particular computation to be performed, a configuration of logic components designed specifically for that computation could be constructed

hardwired programك ⬅ كل عملية حسابية او منطقية (special purpose hardware) محتاجه ترتيب معين من الاسلاك علشان تتنفذ

❑ Suppose we construct a general-purpose configuration of arithmetic and logic functions

❑ This set of hardware will perform various functions on data depending on control signals applied to hardware

❑ With general-purpose hardware, the system accepts data and control signals and produces results

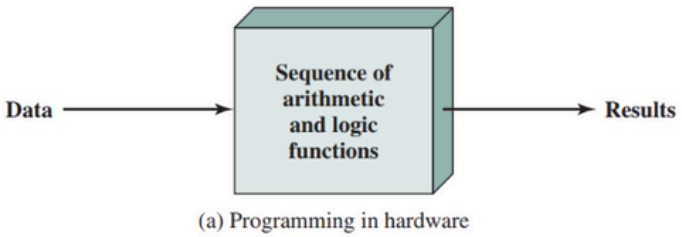
❑ Thus, instead of rewiring the hardware for each new function, the designer merely needs to supply a new set of control signals.

so The new set of control signals is defined by a sequence of commands usually called "Program" يعني بقا سوفتوير مش هاردوايرد(general purpose)

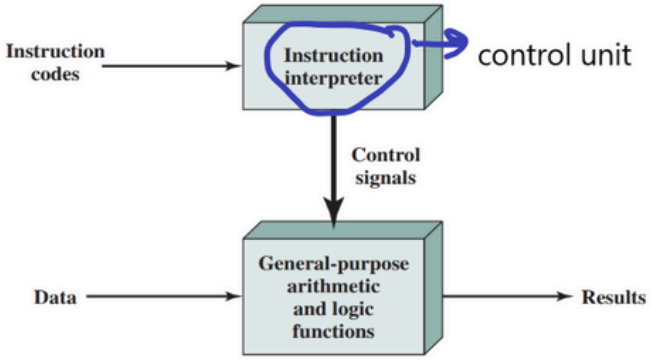
❑ Programming is now much easier.

❑ Instead of rewiring the hardware for each new function, all we need to do is to provide a new sequence of codes.

❑ Each code is an instruction, and there is a part of the hardware interprets each instruction and generates control signals.



(a) Programming in hardware



(b) Programming in software

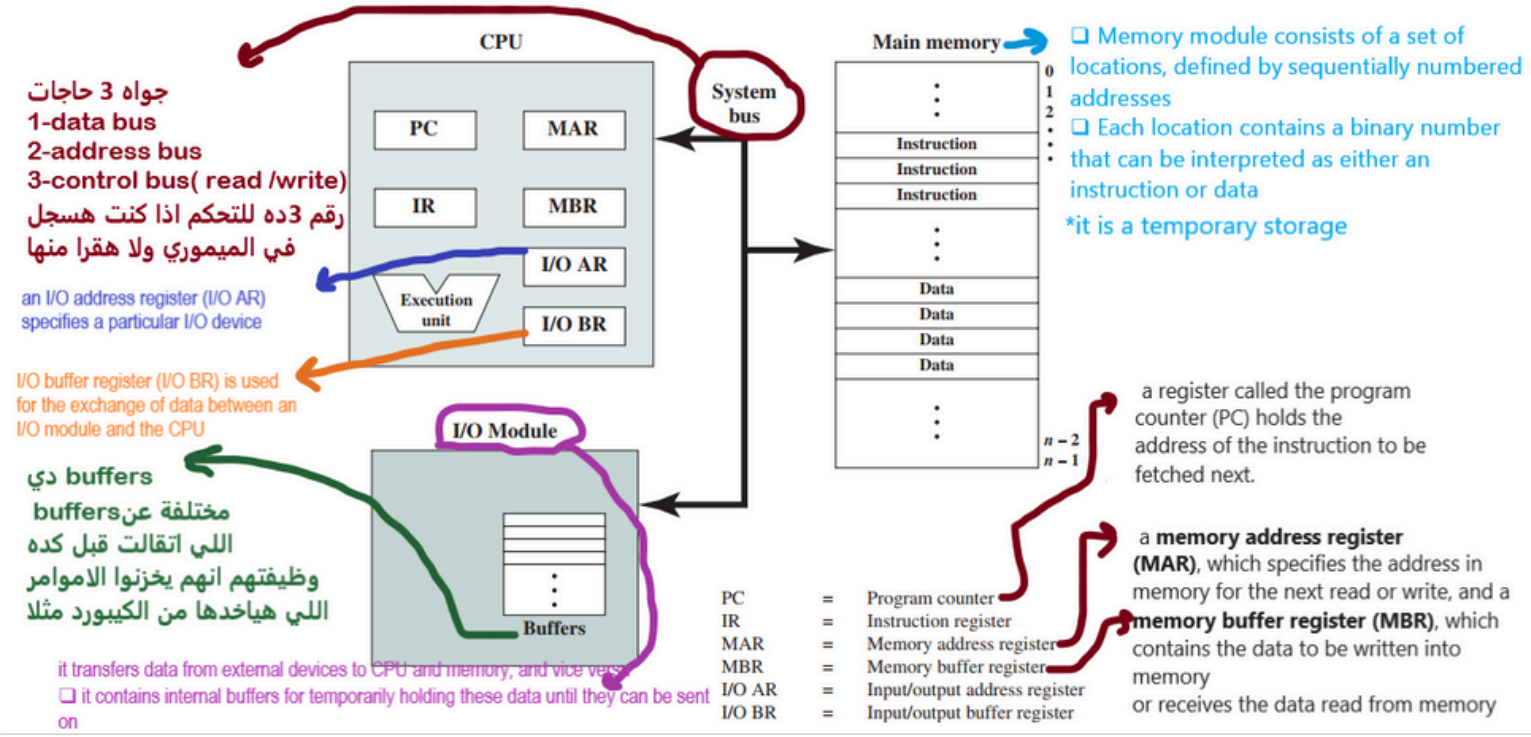
❑ Hardwired systems(hardware) are inflexible

❑ General purpose hardware can do different tasks, given correct control signals(software)

The Program : sequence of steps, for each step an arithmetic or logical operation is done, for each operation, a different set of control signals is needed

⬇ Computer Components

the Central Processing Unit (CPU) consists of The Control Unit and the Arithmetic and Logic Unit.


CPU
PC
MAR
IR
MBR
I/O AR
I/O BR
Execution unit
System bus
Main memory
Instruction
Instruction
Instruction
Data
Data
Data
Data
I/O Module
Buffers
جواه 3 حاجات
1-data bus
2-address bus
3-control bus(read /write)
رقم 3ده للتحكم اذا كنت هسجل في الميموري ولا هقرا منها
an I/O address register (I/O AR) specifies a particular I/O device
I/O buffer register (I/O BR) is used for the exchange of data between an I/O module and the CPU
دي buffers
buffersمختلفة عن
اللي اتقالت قبل كده
وظيفتهم انهم يخزنوا الاموامر
اللي هياخدها من الكيبورد مثلا
it transfers data from external devices to CPU and memory, and vice versa
❑ it contains internal buffers for temporarily holding these data until they can be sent on
❑ Memory module consists of a set of locations, defined by sequentially numbered addresses
❑ Each location contains a binary number that can be interpreted as either an instruction or data
*it is a temporary storage
a register called the program counter (PC) holds the address of the instruction to be fetched next.
a memory address register (MAR), which specifies the address in memory for the next read or write, and a memory buffer register (MBR), which contains the data to be written into memory
or receives the data read from memory
PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register

***instruction cycle (fetch and execute)⬇**

to fetch:

❑Program Counter (PC) holds address of next instruction to fetch

❑Processor fetches instruction from memory location pointed to by PC

❑Increment PC ➢Unless told otherwise

❑Instruction loaded into Instruction Register (IR)

❑Processor interprets instruction and performs required actions

---

to execute:

FORMS OF EXCUTING:

❑Processor-memory ➢Data transfer between CPU and main memory

❑Processor I/O ➢Data transfer between CPU and I/O module

❑Data processing ➢Some arithmetic or logical operation on data

❑Control ➢Altering the execution sequence , e.g : jump operation

> “control (للتوضيح يعني ايه jump)”
> —For example, the processor may fetch an instruction from location 149, which
>
> specifies that the next instruction be from location 182. The processor will
>
> remember this fact by setting the program counter to 182. Thus, on the next
>
> fetch cycle, the instruction will be fetched from location 182 rather than 150

❑Combination of many of the previous forms

Sara Ibrahim

“example of program execution” ”Hypothetical machine”

Assume that ⬇

1-The processor contains a single data register, called an accumulator (AC)

❑ Both instructions and data are 16 bits long thus, it is convenient to organize memory using 16-bit words

❑ The instruction format provides 4 bits for the opcode(operation code)

يعني دلوقتي هم كلهم 16bit هناخد منهم 4 نحدد بيهم الop اذا كانت / add mov.... , وهيفضل 12bit هنسيبهم لتسجيل الaddress

4bit for operations = 2^4 different operations.

12bit for address =2^12 4096 = 4k different word can be addressed in memory .

***remember that 1k =1024 bit**

هفرضلك كمان ان ⬇

0001 👉 load operation (=1 in decimal)

0010 👉store operation (=2 in decimal)

0101 👉add operation (=5 in decimal)

LD = load,(LDA= load A)

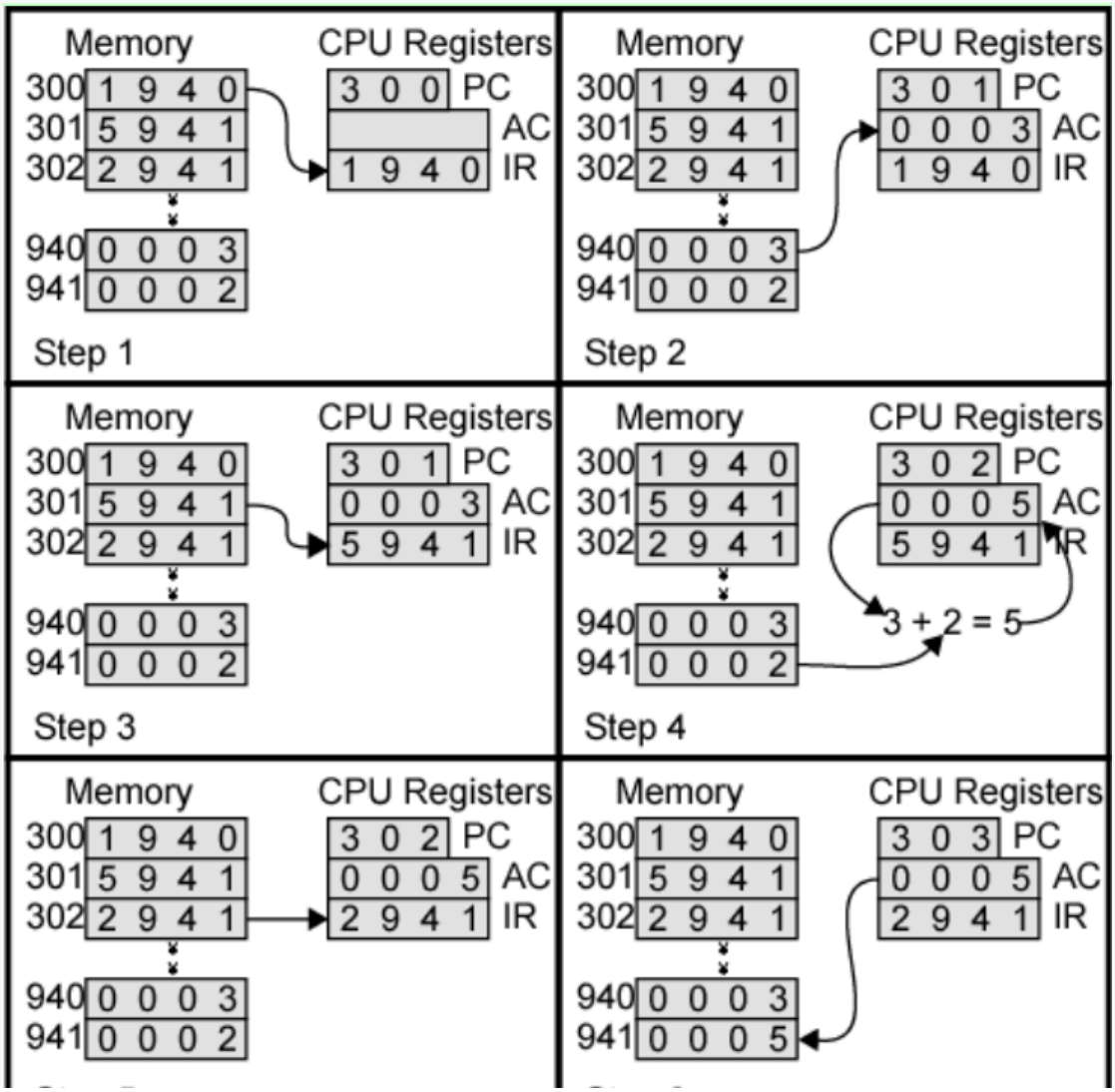
AD=add,(ADA=add to A)

ST= store (STA = store to A)

-you need to execute these instructions:

1-LDA {940} **2- ADA {941}** **3-STA {941}**

فيبقي دي خطوات التنفيذ ⬇



some notes about execution ⬇

❑ With a more complex set of instructions, fewer cycles would be needed

❑ Some older processors, include instructions that contain more than one memory address

❑ The execution cycle for a particular instruction on such processors could involve more than one reference to memory

❑ For example, the PDP-11 processor includes an instruction, expressed symbolically as ADD B,A ⬇

1. Fetch the ADD instruction.

note*In order that the contents of A are not lost , the processor must have at least two registers for storing memory values, rather than a single accumulator

2. Read the contents of memory location A into the processor.

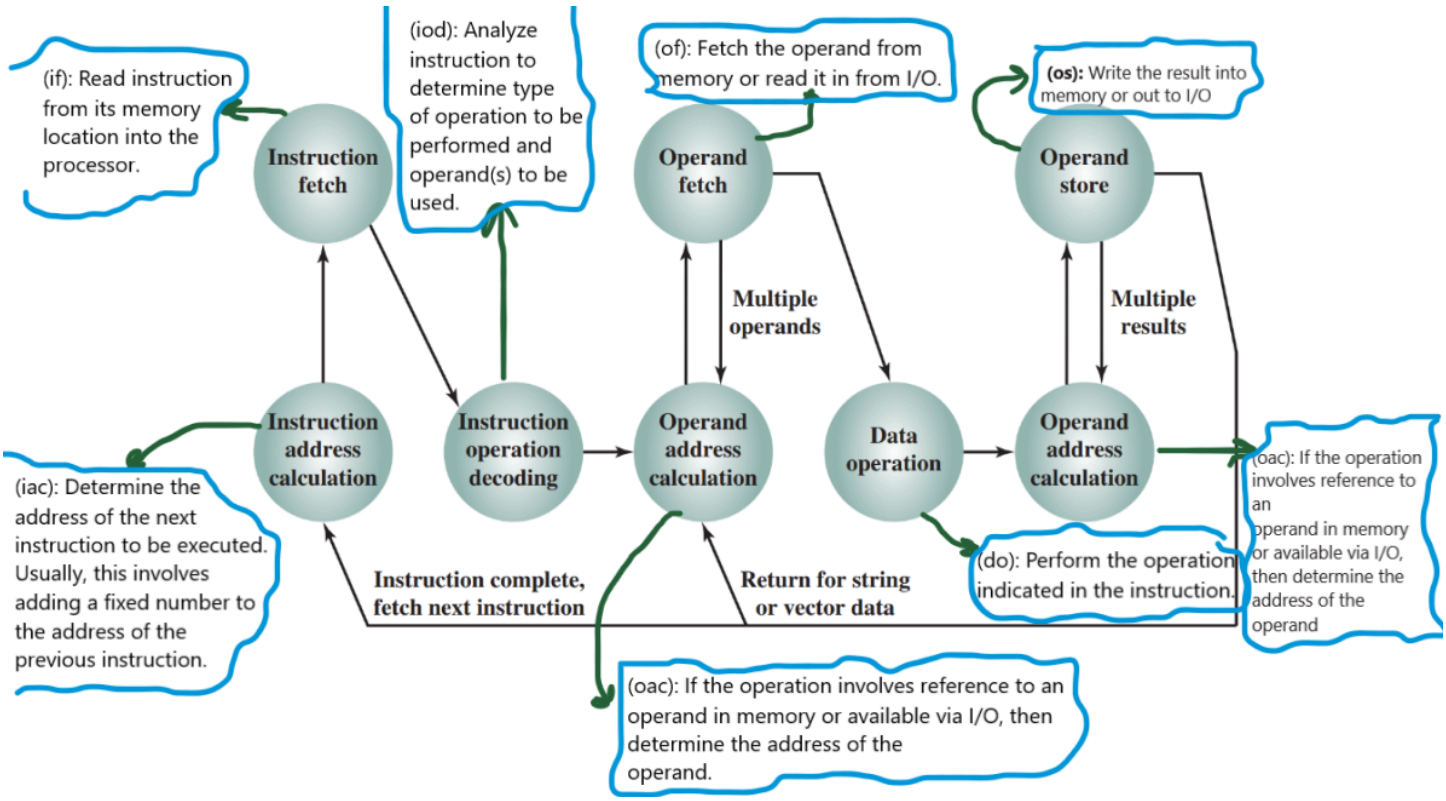
3. Read the contents of memory location B into the processor.

4. Add the two values.

5. Write the result from the processor to memory location A

BUT the execution cycle for a particular instruction may involve more than one reference to memory. Also, instead of memory references, an instruction may specify an I/O operation

With these additional considerations in mind, a more detailed instruction cycle can be discussed ⬇


(if): Read instruction from its memory location into the processor.
Instruction fetch
(iod): Analyze instruction to determine type of operation to be performed and operand(s) to be used.
(of): Fetch the operand from memory or read it in from I/O.
Operand fetch
(os): Write the result into memory or out to I/O
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data operation
Operand address calculation
(iac): Determine the address of the next instruction to be executed. Usually, this involves adding a fixed number to the address of the previous instruction.
Instruction complete, fetch next instruction
Return for string or vector data
(do): Perform the operation indicated in the instruction.
(oac): If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand
(oac): If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.

Interrupts ⬇

❑ Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the CPU

❑ this concept 👉 to understand more clearly the nature of the instruction cycle

❑ We focus on the communication between modules that results from interrupts.

Interrupts are provided primarily as a way to improve processing efficiency.

obviously , most external devices are much slower than the processor. Suppose

that the processor is transferring data to a printer using the instruction cycle scheme. After each write operation, the processor must pause and remain

idle until the printer catches up. The length of this pause may be on the order of

many hundreds or even thousands of instruction cycles that do not involve memory.

Clearly, this is a very wasteful use of the processor

If we make the processor execute instructions during the pause period, this will be more efficient

Q\ But how to notify the processor that the requested I/O operation is completed ?

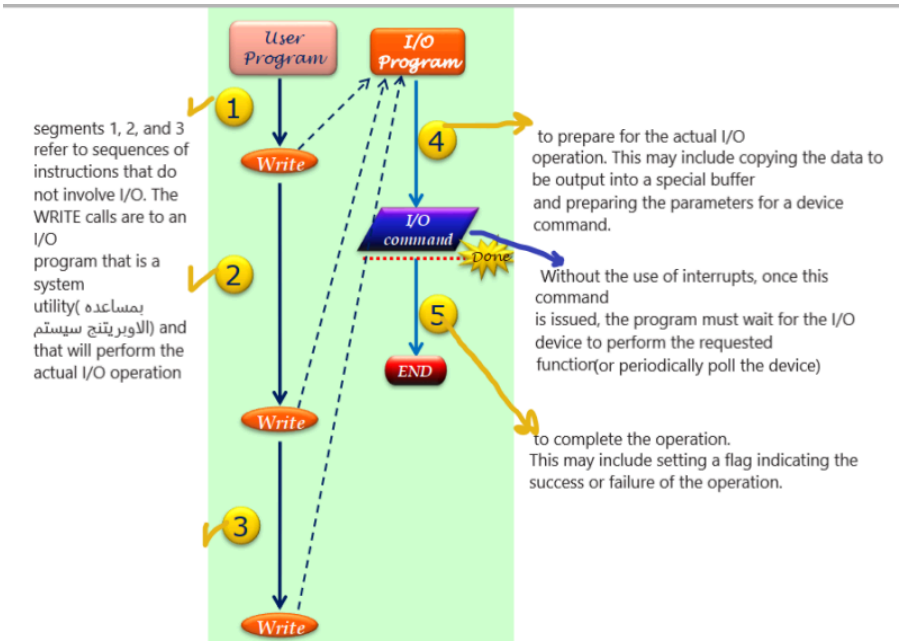
Ans: I/O device send a signal to the processor indicating the completion of I/O request and this signal is called Interrupt signal

❑ It interrupts the normal execution, so processor can finish the I/O operation correctly

لازم ال signal تروح تقول للبروسيسور ان الاوبريشن خلصت علشان يقدر يدخل في اوبريشن تانية بنظام

طيب لو الsignal دي مش عندي؟! هيحصل عشوائية وممكن البروسيسور يدخل في اوبريشن قبل ما اللي قبلها تكون اتنفذت او اصلا ميعملش اي اوبريشن لانه ميعرفش ان فيه اوبريشن هيعملها فيحصل overwrite وافقد الداتا (الكلام ده بيحصل خاصة لو الاوبريشن مع ال network controller)

The user program performs a series of Write calls interleaved with processing ⬇


User Program
I/O Program
1
Write
2
Write
3
Write
4
I/O command
Done
5
END
segments 1, 2, and 3 refer to sequences of instructions that do not involve I/O. The WRITE calls are to an I/O program that is a system utility(يمساعده الاوبريتنج سيستم) and that will perform the actual I/O operation
to prepare for the actual I/O operation. This may include copying the data to be output into a special buffer and preparing the parameters for a device command.
Without the use of interrupts, once this command is issued, the program must wait for the I/O device to perform the requested function(or periodically poll the device)
to complete the operation. This may include setting a flag indicating the success or failure of the operation.

Sara Ibrahim

> **“focus”**
>
> Because the I/O operation may take a relatively long time to complete, the I/O program is hung up waiting for the operation to complete; hence, the user program is stopped at the point of the WRITE call for some considerable period of time.
> — With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.

Interrupt mechanism:

❑ With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress

❑ The I/O program that is invoked in this case consists only of the preparation code and the actual I/O command

❑ This I/O operation is conducted concurrently with the execution of instructions in the user program

❑ When the external device becomes ready to be serviced, the I/O module for that external device sends an interrupt request signal to the processor

❑ The processor responds by suspending operation of the current program, branching off to a program to service that particular I/O device, known as an interrupt handler

❑ Processor resumes execution after the device is serviced

❑ From the point of view of the user program, an interrupt is

just that:

1-An interruption of the normal sequence of execution.

2- When the interrupt processing is completed, execution resumes

❑ Thus, the user program does not have to contain any special

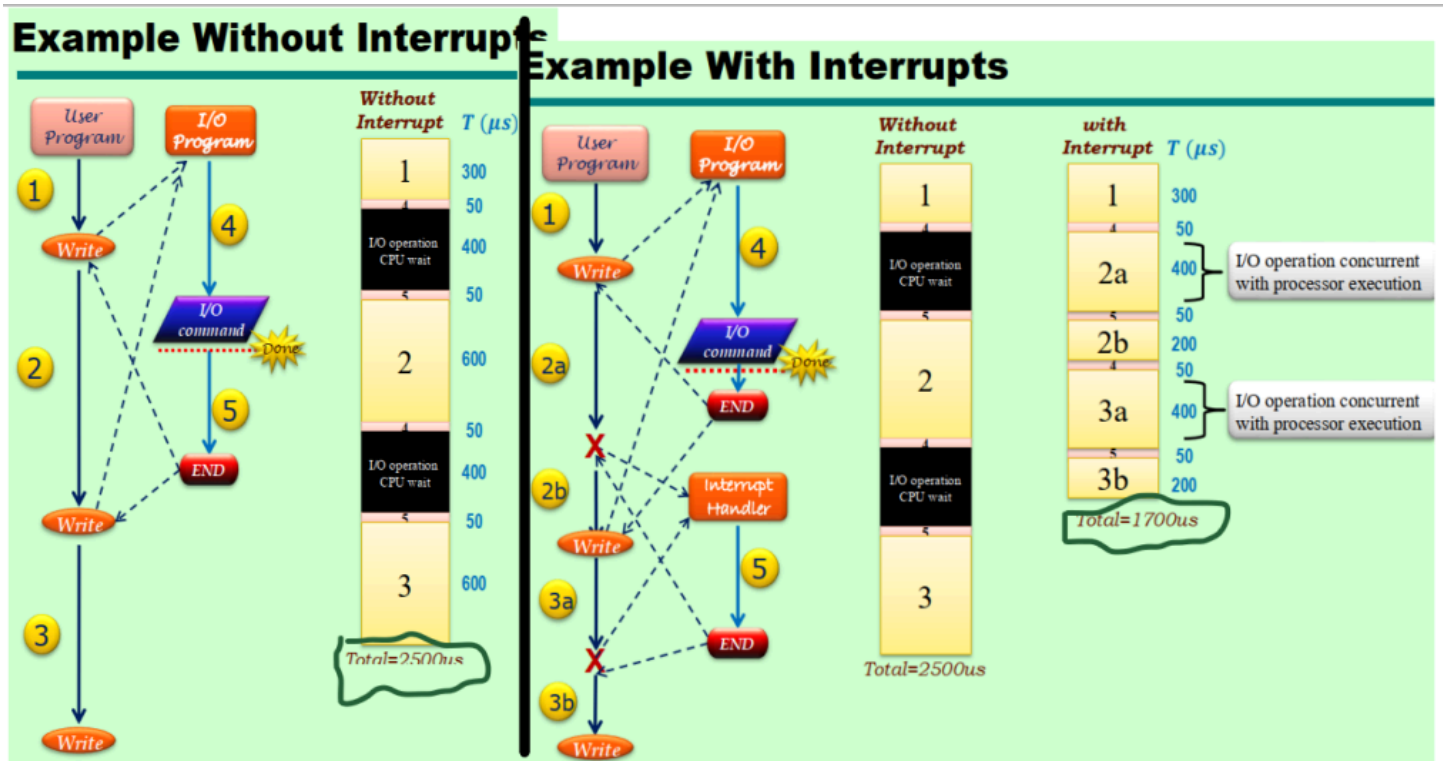
code to accommodate interrupts;

❑ Processor and the operating system are responsible for

suspending user program then resuming it at the same point

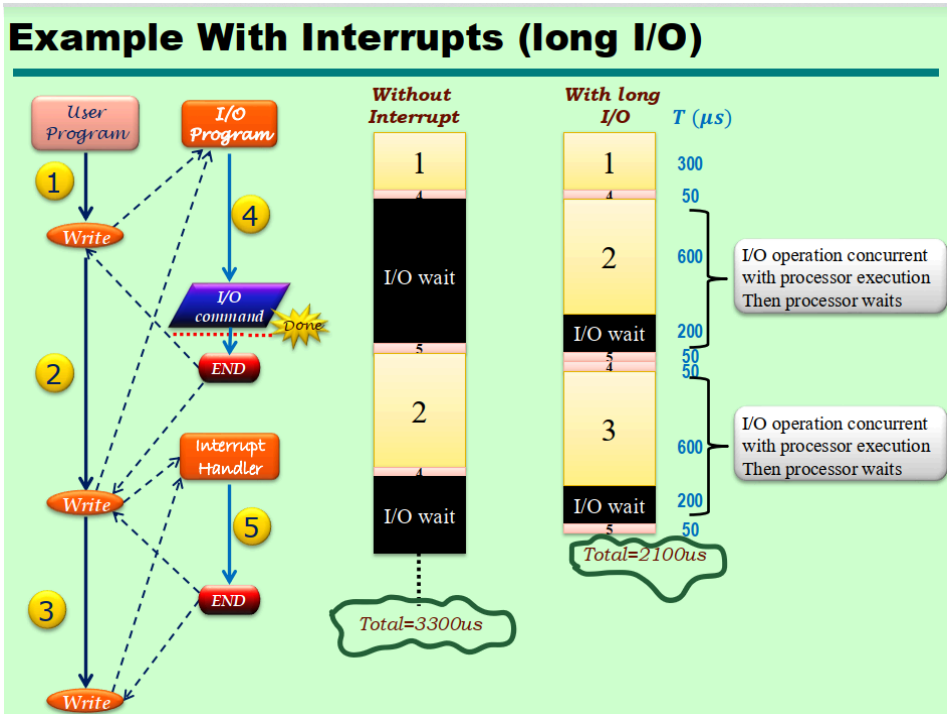
الجدول ده بيوضح كل segment محتاجة وقت قد ايه للتنفيذ ومطلوب نحسب الوقت الكلي للتنفيذ

Code Segment	1	2	3	4	5	I/O wait
Time (μs)	300	600	600	50	50	400



طیب لو کان ال I/O wait کبیر کده ⬇

Code Segment	1	2	3	4	5	I/O wait
Time (μs)	300	600	600	50	50	**800**



Interrupt handler :

❑ It is also called Interrupt Service Routine (ISR)

❑ Typically, this program determines the nature of the interrupt and performs whatever actions are needed.

❑ In our example, the handler determines which I/O module generated the interrupt and may branch to a program that will write more data out to that I/O module.

❑ When interrupt handler routine is completed, the processor resumes execution of user program at the point of interruption

❑ It is clear that there is some overhead involved in this process.

➢ Extra instructions must be executed (in the interrupt handler)

Sara Ibrahim

“ interrupt cycle:

❑ Processor checks to see if any interrupts have occurred,

indicated by the presence of an interrupt signal

❑ If no interrupts are pending, the processor proceeds to the

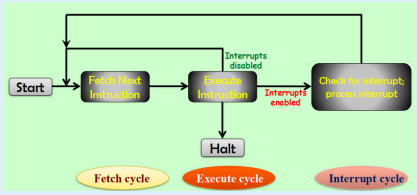
fetch cycle and fetches the next instruction of current program

❑ If an interrupt is pending, the processor does the following:

1- Suspends execution of the current program and saves its context

2- sets the PC to the starting address of an interrupt handler routine

❑ The processor now proceeds to the fetch cycle and fetches the first instruction in the interrupt handler program



”

Table 3.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation, request service from the processor, or to signal a variety of error conditions.
Hardware Failure	Generated by a failure such as power failure or memory parity error.

The communication line controller generates the interrupt

Multiple interrupt :(interrupts are handled in strict sequential order) it is possible for a communications interrupt to occur while a printer interrupt is being processed so This situation is called multiple interrupts

Two approaches to deal with multiple interrupt ⬇

1- disable interrupts while an interrupt is being processed :Thus, when a user program is executing and an interrupt occurs, interrupts are disabled immediately.

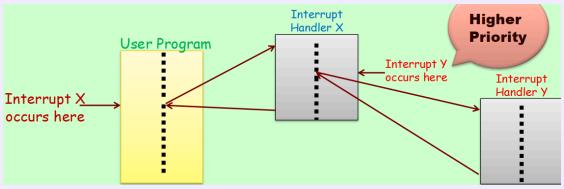
After the interrupt handler routine completes, interrupts are enabled before resuming the user program, and the processor checks to see if additional interrupts have occurred

❑ The drawback : it does not take into account relative priority or time-critical needs

---

2- define priorities for interrupts and to allow

an interrupt of higher priority to interrupt a lower-priority interrupt



I/O function and DMA

I/O instructions similar to memory reference instruction is used to exchange data with I/O device

❑ In some cases, it is desirable to allow I/O exchanges to occur directly with memory.

❑ In such a case, the processor grants to an I/O module the authority to read from or write to memory

❑ During such a transfer, the I/O module issues read or write commands to memory directly.

❑ This operation is known as direct memory access (DMA)

❑ It is a very useful technique for massive data transfer

Chapter 1 👀

1-difference between Organization and Architecture ⬇

Computer architecture refers to those attributes of a system visible to a programmer or, put another way, those attributes that have a direct impact on the logical execution of a program. A term that is often used interchangeably with computer architecture is instruction set architecture (ISA). The ISA defines instruction formats, instruction opcodes, registers, instruction and data memory; the effect of executed instructions on the registers and memory; and an algorithm for controlling instruction execution.

Computer organization refers to the operational units and their interconnections that realize the architectural specifications

> **Examples**
>
> **architectural attributes include the instruction set, the number of bits used to represent various data types (e.g., numbers, characters), I/O mechanisms, and techniques for addressing memory.**
>
> **Organizational attributes include those hardware details transparent to the programmer, such as control signals; interfaces between the computer and peripherals; and the memory technology used**

2- Functions and Structures ⬇

□ Structure: The way in which the components are interrelated.

main four structures :

1- Central processing unit (CPU) 2-Main memory 3- I/O

4-System interconnection(by means of a system bus)

□ Function: The operation of each individual component as part of the structure

Functions :
***Data processing: Data may take a wide variety of forms, and the range of processing requirements is broad.**

***Data storage:**

***Data movement: The computer's operating environment consists of devices that serve as either sources or destinations of data. When data are received from or delivered to a device that is directly connected to the computer, the process is known as input–output (I/O), and the device is referred to as a peripheral.**

note 👉 When data are moved over longer distances, to or from a remote device, the process is known as data communications.

***Control: Within the computer, a control unit manages the computer's resources and orchestrates the performance of its functional parts in response to instructions.**

note :

A hierarchical system is a set of interrelated subsystems, each of the latter, in turn, hierarchical in structure until we reach some lowest level of elementary subsystem. The hierarchical nature of complex systems is essential to both their design and their description. The designer need only deal with a particular level of the system at a time. At each level, the system consists of a set of components and their interrelationships. The behavior at each level depends only on a simplified, abstracted characterization of the system at the next lower level. At each level, the designer is concerned with structure and function

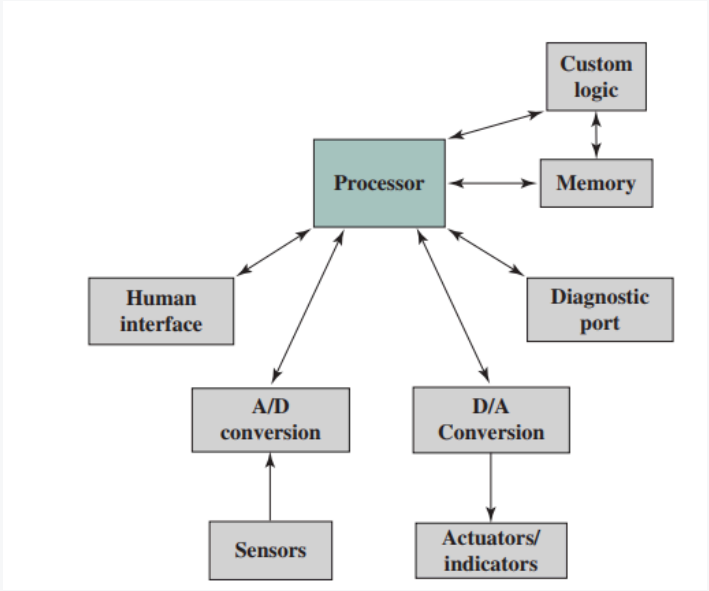
3- Embedded systems ⬇

The term embedded system refers to the use of electronics and software within a product such as all devices except (smart phones , computers)- as opposed to a general-purpose computer a laptop or desktop system-.

**most, devices that use electric power have an embedded computing system. It is likely that in the near future virtually all such devices will have embedded computing systems

باختصار شديد

> الايمبيديد ده نظام اقل امكانيات من نظم الفون او اللاب لان الايمبيديد بيسخدم في اجهزة ابسط
>
> الايمبيديد اهم مكون فيه هي ال microcontroller (اللي في الرسمه تحت)ودي بتبقي محتوية علي كل حاجة وبيكون البروسيسور جزء منها



A microcontroller chip makes a substantially different use of the logic space available

> من ضمن المميزات اللي اقدر اضيفها لاجهزة الايمبيديد اني اوصلها بالنت فتبقي بياناتها متوصلة بسيرفر الشركه المصنعة فلو الجهاز علي وشك العطل تقوم الشركه باعته تحذير للمالك مثلا وهنا تبان فايده IOT

4- IOT

*The Internet of things (IoT) is a term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensor.

*the Internet has gone through roughly four generations of deployment culminating in the IoT:

1. Information technology (IT): PCs, servers, routers, firewalls, and so on.

2. Operational technology (OT): Machines/appliances with embedded IT built by non-IT companies, such as medical machinery,.

3. Personal technology: Smartphones, tablets, and eBook readers bought as IT devices

4. Sensor/actuator technology(IOT): Single-purpose devices bought by consumers, IT, and OT people exclusively using wireless connectivity, generally of a single form, as part of larger systems.

5- Cloud computing ⬇

***Cloud computing: A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.**

***Cloud networking refers to the networks and network management functionality that must be in place to enable cloud computing.**

*** Most cloud computing solutions rely on the Internet, but that is only a piece of the networking infrastructure**

***We can think of cloud storage as a subset of cloud computing**

cloud services

The essential purpose of cloud computing is to provide for the convenient rental of computing resources. A cloud service provider (CSP) maintains computing and data storage resources that are available over the Internet or private networks

Sara Ibrahim